Komputer osobisty IBM-PC zaczyna robić karierę i w naszym kraju. Kilkanaście egzemplarzy tego drogiego, jak na polską kieszeń, sprzętu pracuje już u osób prywatnych. Równocześnie kilka firm podjęło montaż komputerów kompatybilnych z IBM-PC i choć trudno mówić o widokach na szerokie upowszechnienie (jak choćby ZX SPECTRUM), to początek został już zrobiony. Mimo wygórowanych cen zestawu 3,5—7 mln zł), co daje przelicznik ok. 3 tys. zł za dolara — chętnych na „złotówkowego" IBM nie brakuje.

Zamieszczając w mikroKLANIE 13 tekst o normie GKS, zapowiedzieliśmy dalsze publikacje na ten temat. Obecna publikacja, omawiająca uzależnienia sprzętowe i podająca przykładowe procedury graficzne, przygotowuje grunt pod następne artykuły, w których przedstawimy sposób realizacji jądra systemu graficznego zgodnego ze standardem GKS.

Ze względu na objętość tekstu publikujemy go w dwóch częściach: w pierwszej — opis komputera i przykładowe procedury w języku maszynowym, w drugiej — bibliotekę procedur zapisanych w języku C.

Grafika IBM-PC (1)

IBM-PC ma bogate możliwości graficzne. Na monitorze ekranowym można zarówno wyświetlać napisy (tryb alfanumeryczny), jak i rysować (tryb graficzny). Wydruki mogą być monochromatyczne (jasne litery na ciemnym tle) lub kolorowe. Ekran dla trybu alfanumerycznego podzielony jest na 25 linii po 80 lub 40 znaków (do wyboru). W trybie graficznym komputer zapewnia rozdzielczość 640×200 (obrazy czarno-białe) i 320×200 (kolor). Tło może być wybrane spośród 16 kolorów, natomiast same rysunki mogą być wykonywane kolorami z dwóch palet. Na palecie 1 dostępne są trzy kolory: zielony, czerwony i żółty, natomiast na palecie 2 kolory: niebiesko-zielony, fioletowy i biały. Opisana koncepcja grafiki kolorowej wykorzystuje możliwie najmniejszy obszar dla pamięci obrazowej. Kolor każdego punktu na ekranie pamiętany jest w dwóch bitach:
00 — kolor tła
01 — kolor zielony (paleta 1) lub niebiesko-zielony (paleta 2)
10 — kolor czerwony (paleta 1) lub fioletowy (paleta 2)
11 — kolor żółty (paleta 1) lub biały (paleta 2).
W rezultacie cały obraz zajmuje 128 000 bitów (320×200×2), czyli — 16 000 bajtów.

```
        public  clear_image

clear_image :
        push    di
        push    es          ; ochrona rejestrów
        mov     ax,0b800h   ; adres pamięci obrazu
        mov     es,ax
        mov     di,0
        mov     cx,8000     ; ilość słów obrazu
        mov     ax,0        ; kolor tła
        cld
        rep     stosw       ; wpisywanie do pamięci
        pop     es
        pop     di
        ret
```

Wydruk 1. Procedura clear_image w języku ASSEMBLER 8086

Adres początku pamięci obrazu równy jest B0000H dla grafiki czarno-białej lub B8000H — dla kolorowej.

Na wydruku 1 przedstawiono procedurę „zamalowywania" obrazu w trybie grafiki kolorowej 320×200, począwszy od lewego górnego rogu ekranu.

W trybie alfanumerycznym, kody wyświetlanych znaków ASCII pamiętane są w obszarach zaczynających się od tych samych adresów, jak w przypadku trybu graficznego. Dla trybu alfanumerycznego kolorowego z każdym znakiem skojarzony jest bajt opisujący kolor znaku i kolor tła. W efekcie na cały ekran przypada 4000 bajtów (80×25×2) dla trybu 80-znakowego oraz 200 bajtów (40×25×2) dla trybu 40-znakowego.

Bajt opisujący kolor znaku

Bajt								Kolor:
M	T	T	T	I	Z	Z	Z	
	0	0	0		0	0	0	czarny
	0	0	1		0	0	1	niebieski
	0	1	0		0	1	0	zielony
	0	1	1		0	1	1	niebiesko-zielony
	1	0	0		1	0	0	czerwony
	1	0	1		1	0	1	fioletowy
	1	1	0		1	1	0	żółty
	1	1	1		1	1	1	biały

M — migotanie, T — kolor tła, I — intensywność, Z — kolor znaku

```
        public  clear_screen

clear_screen :
        push    di
        push    es          ; ochrona rejestrów
        mov     ax,0b800h   ; adres pamięci ekranu
        mov     es,ax
        mov     di,0
        mov     cx,2000     ; ilość słów
        mov     ax,0720h    ; biały odstęp na czarnym tle
        cld
        rep     stosw       ; wpisywanie do pamięci
        pop     es
        pop     di
        ret
```

Wydruk 2. Procedura clear_screen

Wydruk 2 przedstawia procedurę wypełniania ekranu w trybie alfanumerycznym kolorowym 80×25. Uwaga: kody znaków pamiętane są pod adresami parzystymi, natomiast bajty atrybutów — pod adresami nieparzystymi.

Dysponując podanymi informacjami można już projektować oprogramowanie sterujące monitorem alfanumerycznym i graficznym. Dla uproszczenia programów można wykorzystać procedury zawarte w firmowej pamięci ROM.

Ich wywołanie odbywa się poprzez przerwanie: INT 10H, przy odpowiednio ustawionych wartościach rejestrów.

(cdn)

ANTONI URBAN
Computer Studio Kajkowscy
Gdynia

Po wstępym okresie euforii, prawie każdy użytkownik ZX SPECTRUM ma szczerze dość wykorzystywania magnetofonu jako pamięci masowej. Przyłączenie dysków elastycznych jest jednak rozwiązaniem przewyższającym ceną sam komputer. Dlatego też wiele osób zastanawia się nad alternatywną propozycją firmy SINCLAIR. Sądzimy, że prezentowane artykuły wyjaśnią wiele wątpliwości.

Ponieważ otrzymaliśmy niemal równocześnie dwa teksty na temat ZX MICRODRIVE, zawierające częściowo te same informacje, lecz równocześnie znacznie się różniące, przyjęliśmy dość nietypowe rozwiązanie — chyba z korzyścią dla Czytelników. Z obydwu materiałów wydzieliliśmy podstawowe informacje o ZX MICRODRIVE (akapity wybrane z tekstu Jerzego Karczmarczuka zaznaczamy trójkątem). Jako uzupełnienie publikujemy dwugłos wyrażający opinie autorów o tym urządzeniu.

ZX MICRODRIVE

ZX MICRODRIVE to zewnętrzna pamięć masowa przeznaczona do współpracy z ZX SPECTRUM. Do zapisu danych wykorzystywane są specjalne mikrokasety (ang. cartridge) o pojemności ok. 85—90 KB każda. W kąsetkach znajduje się taśma magnetyczna połączona w pętlę. Długość taśmy wynosi 5 m, a szerokość 1/16 cala. Czas przewijania pełnej pętli — 7 s. Przewijanie odbywa się z tą samą prędkością co zapis i odczyt. ZX MICRODRIVE jest przyłączony do komputera za pośrednictwem wielozadaniowego interfejsu ZX INTERFACE 1, zawierającego system operacyjny zapisany w pamięci EPROM oraz układy obsługujące transmisję danych (MICRODRIVE, ZX NET, RS-232). Ceny (z końca 1984) MICRODRIVE — 50 funtów, ZX INTERFACE 1 — 50 funtów, mikrokaseta — 4,95 funta.

Zapis na kasetce jest dwuścieżkowy; poszczególne bity są zapisywane na przemian. Dane zgrupowane są w sektorach (512 bajtów + nagłówki) — na jednej kasetce mieści się do 200 sektorów, co oznacza gęstość zapisu ok. 500 bajtów na cal. Struktura zapisu jednego sektora jest następująca:

- blok nagłówka (1,25 ms)

— preambuła bloku 12 bajtów (10 bajtów 00H i 2 bajty 0FFH)

— nagłówek sektora 15 bajtów (numer sektora, nazwa kasetki, bajt sumy kontrolnej

- przerwa (3,75 ms)
- blok danych (25 ms)

— preambuła bloku 12 bajtów

— nagłówek rekordu 15 bajtów (nazwa pliku, numer rekordu, długość, bajt sumy kontrolnej

— dane 513 bajtów (dane, bajt sumy kontrolnej)

- przerwa (7 ms).

Dane zajmują więc ok. 2/3 długości taśmy.

Podczas formatowania w każdym sektorze zapisywany jest w bloku nagłówka (z nazwą kasetki i numerem sektora) oraz rekordy z danymi testowymi. Następnie program formatujący sprawdza, ile jest sektorów sprawnych. Sektory, w których wystąpił błąd odczytu zaznaczane są jako „zepsute" i nie są później używane. Stąd wynika różna pojemność mikrokasetek.

▲ Ponieważ formatowanie jest programowane, można skonstruować logicznie mikrokasetę z zupełnie innymi sektorami, o innej długości, bez podziału na dwa bloki, itp. Nawet można trochę skrócić preambułę oraz przerwy między sektorami, są to jednak bardzo ryzykowne operacje i w ich wyniku mikrokasetą może się nadawać do czytania tylko przez jedną fizycznie stację, a i to nie zawsze. Przerwy między blokami służą do pozostawienia procesowi czasu na wykonanie programu podejmującego decyzję co robić ze znalezionym sektorem. Przerwy są za krótkie, by móc przesłać 512 bajtów z (lub do) bufora i jeszcze „złapać" następny sektor. Tak więc dłuższy plik zapisywany jest w co drugim sektorze (jeśli nie musi przeskoczyć przez sektory już zajęte).

▲ Numery sektorów mają znaczenie jedynie pomocnicze. Przy otwieraniu pliku system konstruuje „mapę": tablicę bitów, w której zaznacza, które sektory są wolne. Tablica ta istnieje w pamięci dopóty, dopóki plik jest otwarty. Nigdy nie jest zapisywana na mikrokasecie (zawsze powstaje na bieżąco).

▲ Prędkość transmisji danych przekracza 2 KB/s. Jednak czas dostępu do pliku i czas jego ładowania może być bardzo długi — wskutek nieefektywności oprogramowania zarządzającego.

▲ MICRODRIVE jest przyłączony do ZX INTERFACE 1 następującymi przewodami: 6 przewodów masy, zasilanie 9 V, 2 linie dwukierunkowej szyny danych, 2 linie sterujące wyborem stacji oraz startem—stopem silnika, linia sterująca odczytem—zapisem, linia sterująca kasowaniem oraz linia kontrolna zabezpieczenia mikrokasety przed zapisem (zabezpiecza się przez wyłamanie ząbka w kasetce). Do INTERFACE 1 można jednocześnie przyłączyć szeregowo do ośmiu stacji. Sygnał selekcji stacji jest generowany programowo w pętli osiem razy: siedem razy „nie", raz „tak" (w odpowiedniej sekwencji) i przekazywany wzdłuż łańcuszka MICRODRIVE'ów. Przy programowaniu w języku maszynowym nie ma zabezpieczenia przed włączeniem kilku stacji lub włączeniem stacji bez włożonej kasetki.

▲ Linie sterujące i kontrolne są podłączone do portu o adresie EFH. Wykonując odczyt, program dowiaduje się czy kasetka jest zabezpieczona przed zapisem (można to zignorować!), dalej — jaki jest stan gotowości urządzenia, a także — czy został rozpoznany znacznik początku sektora.

▲ Wysyłając dane do portu o adresie EFH, program włącza silnik, włącza odczyt lub zapis oraz kasowanie (niezależne od zapisu).

▲ Port o adresie E7H jest wykorzystany do transmisji danych. Konwersja szeregowo-równoległa dokonywana jest sprzętowo; program używa po prostu rozkazów blokowej transmisji: INIR i OTIR.

W ZX INTERFACE 1 umieszczone jest 8 K pamięci EPROM, z systemem operacyjnym — w pewnych sytuacjach przesłania ona 16 K pamięci ROM komputera. Nowa pamięć włącza się przy wykonywaniu cyklu pobrania rozkazu z adresu 08H lub 1708H (druga możliwość pozwala „naprawić" błąd w procedurze CLOSE w interpreterze języka BASIC), a wyłącza się przy pobraniu rozkazu z adresu 700H. Rozkazy języka BASIC sterujące ZX MICRODRIVE znajdują się na klawiaturze SPECTRUM — jedak bez przyłączenia ZX INTERFACE 1 przeważnie powodują komunikat o błędzie: Nonsense in BASIC lub też nie dają się w ogóle wykonać.

▲ Wywołanie procedury inicjującej obsługę błędu (RST 08) wymaga przekazania tej procedurze numeru błędu jako parametru umieszczonego w następnym bajcie. Wartość ta dla standardowych błędów wynosi FFH lub mieści się między 00 a 1AH. Inna wartość powoduje wyświetlenie bezsensownego komunikatu. Przy wpiętym ZX INTERFACE 1 błędy 00 — 1AH rozpoznawane jako „zwykłe" i sterowanie wraca do interpretera języka BASIC, jednak kody 1BH do 31H traktowane są jako wywołania procedur w nowym ROM-ie — czytających lub zapisujących jeden sektor mikrokasety, wysyłających informację do RS 232, itp. Kody błędów są traktowane jako indeksy w tablicy procedur mieszczącej się pod adresem 19A9H w nowym ROM-ie. Można to wykorzystać w programach w kodzie maszynowym. Przy powrocie zawsze włącza się stary ROM.

▲ Aby „obejrzeć" nowy ROM, trzeba przez SAVE zapisać jego zawartość na kasetce, a następnie załadować do pamięci roboczej. Można też postąpić w sposób bardziej wyrafinowany. Kod błędu 32 jest opisany jako „zarezerwowany dla Sinclair Research Ltd.". W rzeczywistości powoduje oddanie sterowania procedurze, której adres mieści się w zmiennej systemowej (nowej!/) HD-11 o adresie 23789

(5CEDH), która nominalnie zawiera numer wiersza programu zapisanego z opcją LINE. Umieszczając własny program obsługujący ten „błąd", możemy bez przeszkód kopiować lub oglądać ROM z ZX INTERFACE 1.

System rezerwuje w pamięci RAM 58 bajtów dla zmiennych systemowych (od adresu 23734) oraz dla każdego otwartego zbioru — 32 bajty na mapę sektorów i 595 bajtów na tzw. kanał, którego częścią jest bufor. Zatem początek programu w BASICU przesuwa się „w górę". Jeśli program nie ma miejsca, aby się przesunąć, następuje błąd.

Przykładowo — zapisanie pliku instrukcją **SAVE * „m"; nr" nazwa pliku"** składa się z następujących czynności:

- alokacja kanału i mapy
- włączenie silnika
- przeczytanie wszystkich sektorów ze sporządzeniem mapy bitowej wolnych sektorów i sprawdzeniem, czy nie ma pliku o tej samej nazwie
- przekazanie 9 bajtów opisu pliku do bufora (typ pliku, długość, adres startu itp.); są to informacje, które na kasecie magnetofonowej zapisywane są wraz z nazwą pliku w oddzielnym nagłówku, tutaj stanowią one pierwsze 9 bajtów danych
- przekazanie do bufora bajtów danych; przy każdorazowym wypełnieniu bufora (tzn. przy próbie przekazania 513 bajtu) zapisywany jest nowy sektor, na co składają się następujące czynności:

— odczytanie nagłówka następnego sektora na taśmie
— sprawdzenie w mapie, czy sektor jest wolny
— zapisanie 512 bajtów danych
— zaznaczenie w mapie sektora jako „zajętego"
— zwiększenie numeru rekordu w buforze

- zapisanie ostatniego rekordu; rekord ten ma ustawiony odpowiedni bit w nagłówku oraz z reguły mniejszą liczbę danych
- wyłączenie silnika
- zwolnienie kanału.

▲ Tworzenie plików z danymi wymaga „otwarcia pliku" przez procedurę OPEN, np. OPEN#4,"M";"Dane". Spowoduje to, że w pamięci zostanie zarezerwowany bufor przypisany strumieniowi wyjściowemu nr 4. Instrukcja PRINT#4;... zapisze dane do bufora, a gdy ten się wypełni, zapisze dane na kasetce w pliku o nazwie "Dane". Tak utworzony plik można oczywiście wymazać lub skopiować. Można go czytać przez INPUT#4; lub INKEY$#4; — wymaga to uprzednio zamknięcia pliku przez CLOSE#4 i ponownego jego otwarcia; plik ponownie otwarty można już tylko czytać, nie można go rozszerzać za pomocą standardowych operacji. Strumień nr 2 jest standardowo przypisany ekranowi — najszybszą metodą przejrzenia zawartości pliku jest skopiowanie go na ekran, np. przez MOVE#4 TO#2.

Przykładowe rozkazy języka BASIC do współpracy z ZX MICRODRIVE:

FORMAT "m";1;"biblioteka" — formatowanie kasetki, napęd nr=1, nazwa kasetki="biblioteka"

CAT 1 — spis plików z podaniem wolnej pamięci w KB

ERASE "m";1;"dane" — skasowanie plików "dane"

SAVE* "m";1;"prog1"LINE 10 — zapisanie programu "prog1" — tak, aby po wczytaniu automatycznie startował od linii nr 10

LOAD* "m";1;"kod1"CODE 32768 — wczytanie zbioru "kod1" od adresu 32768

RUN — wczytanie i zastartowanie programu run"

VERIFY* "m";1;"tablica2"DATA a() — sprawdzenie, czy plik zawiera te same wartości co tablica a ( )

MERGE* "m";1;"prog2" — dołączenie do programu istniejącego — programu zapisanego pod nazwą "prog2"

OPEN 4;"m";1;"spis" — przyporządkowanie do kanału nr 4 pliku „spis"; do tak otwartego pliku możliwy jest dostęp sekwencyjny za pomocą instrukcji PRINT#4, INPUT#4, LIST#4 i INKEY$#4

MOVE#4 TO#15 — przepisanie pliku przyporządkowanego do kanału nr 4 — do zbioru przyporządkowanego do kanału nr 15.

Pliki zapisane instrukcją SAVE mogą być odczytane tylko instrukcjami LOAD, MERGE i VERIFY. Zbiory zapisane przez PRINT#, LIST# i MOVE mogą być odczytane tylko za pomocą INPUT#, INKEY$# i MOVE.
System operacyjny pozwala na rozszerzenie języka BASIC o nowe instrukcje. Jeżeli dana instrukcja nie daje się zinterpretować ani przez „nowy" ani przez „stary" ROM, to wykonywany jest jeszcze skok do procedury, której adres zapisany jest w RAM w zmiennej systemowej VECTOR (adres: 23735).

▲ W zbliżony sposób można rozszerzać zestaw instrukcji języka BASIC i bez ZX INTERFACE 1 — nawet dla ZX 81. Procedura obsługi błędu pobiera ze zmiennej systemowej ERR-SP adres komórki stosu, która zawiera adres restartu: procedury drukującej diagnozę i wracającej do głównej pętli edytora języka BASIC. Można wymienić adres restartu lub zawartość ERR-SP, wymaga to jednak dobrej znajomości systemu ZX.

Opracowano na podstawie materiałów nadesłanych przez:
JERZEGO KARCZMARCZUKA (▲)
z Krakowa oraz
TADEUSZA WILCZKA i TOMASZA ZIELIŃSKIEGO
z Warszawy

*

Nie wszystkie programy z kaset magnetofonowych dają się bezpośrednio przenieść na MICRODRIVE. Jeżeli program zostawia mało miejsca na BASIC (tzn. obniża znacznie RAMTOP), to może dojść do sytuacji braku pamięci na zmienne systemowe i kanały (potrzebna rezerwa — ponad 600 bajtów). Ponadto segmenty kodu maszynowego zapisane np. w instrukcjach REM będą się zaczynać — po przesunięciu początku pamięci zawierającej program w języku BASIC — od innego adresu. Ich uruchomienie powoduje z reguły zawieszenie systemu.

Mikrokasety są niestety nośnikiem zawodnym. Zdarza się, że niektóre rekordy po nagraniu i nawet po zweryfikowaniu nie dają się po pewnym czasie odczytać. Należy się przed tym zabezpieczać nagrywając ważniejsze zbiory równolegle na dwie kasetki oraz od czasu do czasu robiąc kopie na kasecie magnetofonowej. Zbiory po nagraniu należy weryfikować (nie dotyczy to zbiorów sekwencyjnych zapisanych przez instrukcje PRINT#, LIST# i MOVE, przy których możliwości tej nie ma).
Błąd w sytemie oprercyjnym spowodował, że jeżeli podczas wykonywania instrukcji SAVE zabraknie pamięci, to system zawiesza się przy włączonym silniku. Instrukcja obsługi zabrania wyjmowania kasetki lub odłączania zasilania, jeżeli ZX MICRODRIVE pracuje. Środkiem zaradczym jest tu krótki (8 bajtów) program, sprawdzający przed rozpoczęciem wykonywania instrukcji liczbę wolnych bajtów.

ZX MICRODRIVE stanowi świetną pomoc szczególnie w programowaniu w kodzie maszynowym. Podczas testowania programów dochodzi często do zawieszania się systemu. Powtórne wczytanie programów ASSEMBLER i DEBUGGER oraz tekstu źródłowego trwa tylko kilkanaście sekund. Czas zapisu zbioru znacznie wzrasta, jeżeli na kasetce jest mało wolnego miejsca.

Istnieje już dzisiaj wiele programów przystosowanych do współpracy z ZX MICRODRIVE. Są to m.in. programy użytkowe: DECPAC (asembler, debuger i disasembler), PASCAL, MASTERFILE (baza danych), TASWORD (procesor tekstu), OMNICALC II (program kalkulacyjny), BETA BASIC (rozszerzenie BASICA), CASH CONTROLLER (budżet domowy).

Programy współpracują z ZX MICRODRIVE w ten sposób, że wczytuje się do pamięci cały zbiór, który po przetworzeniu zapisuje się z powrotem na kasetce. Zazwyczaj nie stosuje się tutaj współpracy interakcyjnej, cha-

rakterystycznej dla dysków elastycznych; poważnym ograniczeniem jest tu czas dostępu i zawodność.

Duże zalety ZX MICRODRIVE w porównaniu z magnetofonem ujawniają się w przypadku zbiorów często modyfikowanych — takich jak kartoteki czy budżet.

Pamięć ZX MICRODRIVE jest cennym uzupełnieniem mikrokomputera ZX SPECTRUM, podnoszącym jego walory użytkowe. Nie bez znaczenia jest tutaj stosunkowo niska cena stacji i interfejsu.

TADEUSZ WILCZEK
TOMASZ ZIELIŃSKI

*

MICRODRIVE ma przed sobą przyszłość (choć chyba nie w wersji proponowanej przez SINCLAIRA, dzięki swojej niebywałej prostocie i — co za tym idzie — niskiej cenie. Aktualna cena jest po prostu nieprzyzwoita: urządzenie to jest konstrukcyjnie prostsze od przeciętnego magnetofonu kasetowego. Korzysta z głowicy niczym nie różniącej się od typowej głowicy stereo. Mechanika jest bardzo prosta, elektronika też. A trzeba przy tym pamiętać że stacja dysków 5.25" do SPECTRUM (Viscount) kosztuje 245 funtów.

Bardziej skomplikowana jest konstrukcja mikrokasety. Jednak jej cena również jest zawyżona; wszyscy fachowcy twierdzą, że spadnie, choć niestety — nikt nie wie kiedy.

INTERFACE 1 wykorzystuje porty we—wy o adresach E7H, EFH i F7H, tj. trzecią i czwartą linię adresową. Może to być źródłem kłopotów w przypadku używania razem z INTERFACE 1 niestandardowych urządzeń.

Po około trzech miesiącach dość intensywnej pracy nie miałem ani jednego przypadku zauważalnego przekłamania, nie mówiąc o uszkodzeniu taśmy (odpukać!). Na jednej mikrokasecie mam jeden uszkodzony sektor, którego użycie powodowało, że po kilkudziesięciu sekundach kręcenia taśmą komputer sygnalizował „File not found", mam jednak podstawy sądzić, że ten fragment taśmy nie zepsuł się w trakcie eksploatacji, lecz był od początku zły[1] — być może jest to miejsce klejenia tak niefortunnie wykonane, że prymitywna procedura formatująca przeoczyła błąd. Gorzej, że taki sektor — zepsuty, ale prawidłowo sformatowany — będzie przeszkadzał i trzeba go unieszkodliwić wpisując plik „manekin" za pomocą specjalnie w tym celu napisanego programu.

Moje główne problemy? Jednak strach przed uszkodzeniem taśmy, co powoduje, że wszystkie ważne dane i tak kopiuję na kasetę. Dalej — stosunkowo mała pojemność: moje mikrokasety mają po 90 K pamięci. Wreszcie — powolność odczytu—zapisu i niewygodne korzystanie z plików.

Wczytywanie pliku za pomocą standardowych procedur ładuje sektory o kolejnych numerach. Jeśli plik był wpisany na dość wypełnioną taśmę i sektory zostały umieszczone w kolejności niezbyt naturalnej, to zanim cały plik zostanie wczytany — MICRODRIVE kręci taśmą aż do znudzenia. Oczywiście należało to zorganizować rozsądniej.

Po zapisaniu pliku użytkownik zamyka go. System zapisuje ostatni rekord wraz ze specjalnym znacznikiem końca. Jeśli wskutek błędu powstał na taśmie plik bez znacznika końca, nie dzieje się nic specjalnie złego: wymazywanie takiego pliku oraz próba wczytania nie istniejącego rekordu trwa kilkadziesiąt sekund. Mam na to własne określenie: Młynek Modlitewny Sir Clive'a...

Zapisanego i zamkniętego pliku nie da się rozszerzyć inaczej jak tylko przez skopiowanie całości pliku na nowy i dopisanie doń informacji. Jest to prymitywne i niewygodne rozwiązanie. Nie ma żadnych przeszkód, aby skonstruować znacznie lepszy system obsługi ZX MICRODRIVE.

Można mieć pliki indeksowane albo adresowalne bajtowo. Można prosto napisać program rozszerzający plik; wystarczy wczytać ostatni rekord i wymazać zeń znacznik końca pliku. Można posłużyć się bardziej efektywnymi algorytmami rozmieszczającymi informację na mikrokasecie i optymalizującymi dostęp do plików. Wszystkie te udogodnienia producent zostawił dla użytkowników i niezależnych twórców oprogramowania.

W zamian producent wprowadził „zabezpieczenia", które np. uniemożliwiają MERGE programów zapisanych opcją LINE i nie pozwalają czytać zawartości plików zapisanych przez SAVE* — można je tylko ładować przez LOAD*. Miało to na celu ochronę programów komercyjnych. Oczywiście, nawet dla średnio wykwalifikowanych piratów nie są to duże zabezpieczenia. Co więcej, kopiowanie kasetki jest łatwiejsze niż kasety magnetofonowej (przez komputer, a nie bezpośrednio), gdyż można kopiować sektor po sektorze, a program na kasecie magnetofonowej stanowi fizyczną i logiczną całość.

Warto tu wspomnieć, że samostartujący program zapisany na kasecie również można zabezpieczyć przed załadowaniem przez MERGE, jeśli zawiera on jeden wiersz w BASICU, który został „nadpsuty" przez wpisanie bardzo dużej liczby w pole zawierające jego długość [1]).

Czy zachęcać do kupna MICRODRIVE? Zależy kogo. Nabywca musi zdawać sobie sprawę, że na razie oprogramowanie nie jest najlepsze. ROM w INTERFACE 1 zawiera błędy, choć nieznaczne moim zdaniem — w porównaniu np. z błędem psującym bity zaokrąglenia w operacji dzielenia, mogących jednak wpłynąć na opóźnienie decyzji kupna; a nuż późniejsze wersje będą poprawione? Tyle, że drobny błąd w dzieleniu był sygnalizowany już na początku kariery ZX81, a mimo to algorytmy arytmetyki zmiennoprzecinkowej zostały przepisane na SPECTRUM praktycznie bez zmian...

Niezależne firmy zajmujące się oprogramowaniem dopiero zaczynają oferować programy dostosowane do ZX MICRODRIVE, a ich dotarcie do Polski jeszcze się opóźni. Aktualnie najbardziej rozpowszechnione wersje PASCALA HISOFT: HP4S i HP4M1.6 nie pozwalają efektywnie korzystać z ZX MICRODRIVE. Co prawda można napisać własny program w kodzie i dołączyć do tekstu źródłowego, ale nie można pod kontrolą edytora ani kompilatora korzystać z podprogramów na mikrokasecie. HISOFT jednak stale ulepsza swoje produkty.

Najpopularniejsze wersje FORTHA: firmy ARTIC oraz znacznie wygodniejszy firmy ABERSOFT — również ignorują MICRODRIVE. Tu jednak, ze względu na specyfikę FORTHA, raz napisane rozszerzenie systemu można na stałe w nim „zamrozić" i jest to o wiele łatwiejsze niż próba przeróbki kompilatora PASCALA!

W stosunkowo dobrej sytuacji są programy, które korzystały z magnetofonu za pośrednictwem języka BASIC — wystarczy wtedy wymienić SAVE na odpowiednie SAVE*, itp; oczywiście, o ile jest miejsce na zmienne systemowe i kanał. Tak postępuje — na przykład — prosty, ale dość użyteczny procesor tekstów TASWORD 2.

Polecałbym więc zakup ZX MICRODRIVE osobom, które dobrze poznały komputer, nie boją się pisania programów systemowych, a nie mają możliwości kupna znacznie droższych dysków. Nawet niezbyt zaawansowani programiści, ale piszący bardzo długie programy lub ciągi danych, które trzeba wpisywać na raty, na pewno odetchną po wątpliwych przyjemnościach żonglowania kasetami magnetofonowymi.

JERZY KARCZMARCZUK

LITERATURA

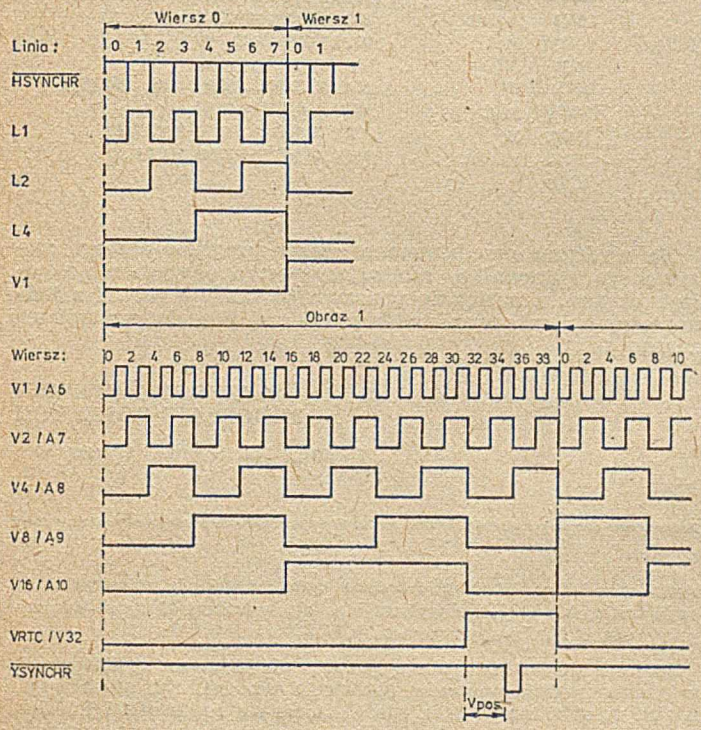
[1] Logan I.: Spectrum Microdrive Book. Melbourne Hause, 1983
[2] Your Computer. No. 12/83, 6/84
[3] ZX SPECTRUM Microdrive and Interface 1 manual.

[1]) Za ten chwyt pragnę podziękować Pawłowi Mąćków z Wrocławia — przypis autora

Pierwsza część artykułu ukazała się w poprzednim numerze. Drugą częścią kończymy opis sterownika zrealizowanego na układach TTL. Aby ułatwić Czytelnikom wykorzystanie układu, zamieszczamy również program obsługi. Niebawem opiszemy zintegrowany sterownik CRT.

Alfanumeryczny sterownik monitora telewizyjnego (2)

Impulsy synchronizacji poziomej HSYNCHR zliczane są przez liczniki U9 i U10. Wyjścia L1, L2 i L4 (U9) podają numer wyświetlanej linii i adresują odpowiednio pamięć ROM (U1). Wyjścia V1...V16 określają numer wyświetlanego wiersza znaków i dołączone są do wejść adresowych $A_6...A_{10}$ pamięci RAM. Po zliczeniu 32 wierszy (rys.) wytwarzany jest impuls VRTC (ang. verical retrace), wygaszający plamkę na czas powrotu do lewego górnego rogu ekranu. Okres trwania impulsu VRTC jest równy czasowi wyświetlania ośmiu wierszy. Gdy stan licznika wierszy osiąga wartość 32+8, zostaje on wyzerowany (wejścia zerujące R_{01} i R_{02} połączone są z wyjściami V32 i V8) i liczenie wierszy rozpoczyna się od początku. Przednie zbocze impulsu VRTC wyzwala uniwibratory U12 (74123), wytwarzające impuls synchronizacji pionowej VSYNCHR/. Położenie tego impulsu w przedziale czasowym, przeznaczonym na powrót plamki jest regulowane potencjometrem V. Umożliwia to właściwe ustawienie położenia obrazu w pionie.



Przebiegi czasowe zespołu liczników linii i wierszy

Całkowita liczba linii obrazu (łącznie z liniami przypadającymi na czas powrotu plamki wynosi 8×(32+8)=320 linii, co nieznacznie różni się od przyjętej w standardzie telewizyjnym liczby 312,5 linii na obraz (bez wybierania międzyliniowego). Przy czasie trwania linii równym 64 µs częstotliwość wyświetlania obrazów wynosi 48,8 Hz, zamiast wymaganej w standardzie TV częstotliwości 50 Hz.

Impulsy synchronizujące HSYNCHR/ i VSYNCHR/ są sumowane z sygnałem wizyjnym (otrzymywanym z rejestru przesuwnego) w obwodzie wejściowym tranzystora T1, pracującego w układzie wtórnika emiterowego. Rezystory R_1 i R_2 ustalają właściwy stosunek poziomu sygnału wizyjnego i impulsów synchronizacji. Z obwodu emitera T1 wyprowadzany jest zespolony sygnał wizyjny. Potencjometr R_4 umożliwia regulację poziomu tego sygnału. Kondensator C_1 oddziela obwód emitera T_1 od ewentualnej składowej stałej, jaka może wystąpić na wejściu wizyjnym monitora. W zależności od typu monitora, może być wymagana różna biegunowość tego kondensatora.

Sygnały wygaszające plamkę na czas powrotu poziomego i pionowego (HRTC i VRTC), są sumowane w jeden wspólny sygnał wygaszający VSP/ (ang. video suppression). Jest on wpisywany do przerzutnika 7474 (U13) impulsem LOAD/, ładującym również rejestr przesuwny U25. Konieczność takiego synchronizowania sygnału VSP/ (a w szczególności sygnału HRTC) impulsem LOAD/ wynika ze sposobu wykorzystania rejestru. Sygnały składające się na linię danego znaku wprowadzane są do rejestru pod koniec cyklu wyświetlania poprzedniego znaku i wyprowadzane z rejestru do wejścia wizyjnego w cyklu adresowania następnego znaku (por. rys. 2 w pierwszej części). Doprowadzenie sygnału HRTC (VSP/) do wyjścia wizyjnego musi być więc opóźnione o jeden znak. Otrzymywany z wyjścia przerzutnika U13 sygnał VSP/ DEL blokuje przejście dla sygnałów z rejestru przesuwnego i wymusza wygaszanie ekranu na czas trwania impulsów HRTC i VRTC.

Rozdzielanie poszczególnych znaków od siebie — zarówno w pionie, jak i w poziomie — zapewnione jest przez postać znaków umieszczonych w ROM-ie (U1). Użyty w sterowniku krajowy układ MCY 7304AA pozostawia zawsze niezaświetloną pierwszą linię i ostatnią kolumnę pola znaku. Efektywne pole znaków ma wymiary 5×7 punktów.

Kursor realizowany jest w postaci migających dwóch najniższych linii pola znaku i wyświetlany jest w przypadku wpisania jedynki w najbardziej znaczącym bicie bajtu znaku w pamięci obrazu. Wyprowadzany z pamięci RAM sygnał bitu D7 jest synchronizowany impulsem LOAD/ za pomocą przerzutnika U13 — dla uzyskania opóźnienia o jeden znak (podobnie jak dla sygnału VSP/). Uzyskany w ten sposób sygnał D7 RAM DEL jest doprowadzany do wejścia bramki U14 (7430), formującej sygnał włączenia kursora. Bramka U14 mnoży logicznie D7 RAM DEL z sygnałami wyjściowymi L2 i L4 licznika linii i wysyła sygnał włączający kursor tylko w czasie wyświetlania linii 6 i 7 (L2=L4=1). Bramka jest blokowana sygnałem LOAD/ w czasie wyświetlania ostatniego (nieaktywnego) punktu znaku oraz sygnałem CSP/ — na czas powrotów plamki. Sygnał BLINK, uzyskiwany w wyniku podzielenia sygnału BRTC przez licznik U15, powoduje migotanie kursora z częstotliwością ok. 3 Hz. Sygnał CURSOR/ sumowany jest z sygnałem wizyjnym, w sposób wymuszający rozświetlenie ekranu bez względu na poziom sygnału wysyłanego przez rejestr przesuwny. Kursor może zostać zredukowany do najniższej (siódmej) linii pola znaku w przypadku dołączenia do wejścia U14 dodatkowo sygnału L1.

Pamięć obrazu (RAM U16...U19) jest w czasie wyświetlania adresowana przez zespół liczników znaków i wierszy (sygnały H1...V16) w wyniku ustawienia sygnału MUX=1 przełączającego multipleksery U4...U6. Pamięć jest w tym czasie utrzymywana w stanie aktywnym i pracuje w trybie odczytu (WE/=1). Pamięć może być dołączona do systemu mikroprocesorowego przez przełączenie multiplekserów linii adresowych (sygnał MUX=0) oraz uaktywnienie dwukierunkowego bufora szyny danych U20, U21. Dostęp do pamięci możliwy jest jedynie w czasie wygaszania plamki. Dołączanie pamięci do systemu mikroprocesorowego w czasie wyświetlania obrazu zaburza bowiem proces wyświetlania i powoduje występowanie na ekranie wielu dodatkowych, chaotycznie migocących punktów świetlnych.

Możliwość dostępu do pamięci wyznacza sygnał BLK/—VSP=1, wyłączający plamkę. Sygnał ten ustawia jednocześnie MUX=0, dołączając linie adresowe pamięci obrazu do szyny adresowej systemu mikroprocesorowego oraz odblokowuje sygnały uaktywniające pamięć RAM, bufor szyny danych i strob zapisu do pamięci. W przypadku czytania pamięci, sygnał MEMR/ zmienia kierunek przenoszenia sygnałów przez bufor danych (DIEN/ 8216=0).

W czasie wyświetlania, sygnał BLOK/—VSP=0 blokuje dostęp mikroprocesora do pamięci U16...U19, wymuszając MUX=1, CS/ (8216)=1 i WE/=1. W przypadku zaadresowania w tym czasie pamięci obrazu zgłaszanego przez dekoder adresowy (U24) sygnałem MEMSEL=1, praca procesora jest wstrzymana do chwili zakończenia wyświetlania linii przez wysłanie sygnału braku gotowości do współpracy WAITRQ/—READY=0. Sygnał braku gotowości musi być wysłany z wyprzedzeniem, gwarantującym zatrzymanie każdego cyklu dostępu do pamięci, który mógłby być zakończony po rozpoczęciu wyświetlania linii. W tym celu praca procesora jest wstrzymywana już w przypadku próby uzyskania dostępu do pamięci ekranu w czasie ostatniego cyklu znaku przebiegu HRTC. Bramka, wysyłająca sygnał WAITRQ/—READY przy MEMSEL=1, jest blokowana sygnałem wyjściowym Q przerzutnika U22. Przerzutnik ten zerowany jest przez przednie zbocze impulsu HRTC (podanego na wejście T), a następnie ustawiany (Q=1), gdy sygnały wyjściowe H1...H16 liczników znaków są wszystkie w stanie 1, co występuje po zliczeniu 95 znaków (jeden znak przed końcem impulsu HRTC — (p. rys.). Stan H1...H16=1 dekoduje bramka U23, dołączona do wejścia S/ przerzutnika. Bramka ta jest blokowana sygnałmi HRTC i VRTC, co uniemożliwia wysłanie sygnału WAITRQ/ w czasie wygaszania plamki.

Dekoder adresu pamięci obrazu (U24) ustawia MEMSEL==1 przy A11...A15=1. Ustalono w ten sposób adresy pamięci obrazu w zakresie 0F800H...0FFFFH. Przez dołączenie dodatkowego inwertera (NOT) negacji między punkty a—b linii A15, można je zmienić na 7800H...7FFFH.

Dekoder adresowy powinien być bramkowany strobami dostępu do pamięci MEMR/ i MEMW/ dla uniemożliwienia wykrywania przez dekoder zbieżnych adresów urządzeń we-wy i wysyłania zbędnych sygnałów WAITRQ/—READY. Stosowanie takiego bramkowania jest jednak niemożliwe w systemach z procesorem 8080 i kontrolerem 8228 (zbyt późno wysłany sygnał MEMW/ nie umożliwia wstrzymania cyklu zapisu do pamięci sygnałem WAITRQ/ generowanym po odebraniu strobu MEMW/). Takie bramkowanie dekodera jest natomiast możliwe w przypadku stosowania kontrolera 8238 (lub w systemach z innym procesorem, np. Z80 — sygnałem MEMRQ). W systemach z procesorem 8080 i kontrolerem 8228 należy natomiast blokować dekoder adresowy sygnałami I-OR/ i I-OW/.

Pamięć obrazu (2 KB) złożono z czterech układów typu 2114, o organizacji: 1024 bity×4. Pamięć ta może być także zbudowana z układów 8102 (MCY 7102) o organizacji: 1024 bity×1; jednak konieczne jest wówczas użycie 16 takich układów. Dodatkowo, należy połączyć rozdzielone wejścia—wyjścia tych pamięci odpowiednio z wyjściami DO i wejściami DI buforów danych 8216 i wejściem pamięci ROM.

Jako generatora znaków użyto ROM typu MCY 7304AA (inna ostatnia litera symbolu oznacza generator znaków programowany nie w kodzie ASCII!), realizujący przetwarzanie 64 podstawowych znaków ASCII (cyfry, duże litery i znaki pomocnicze) o kodach od 20H do 5FH. Zamiast tego układu można zastosować odpowiednio zaprogramowany EPROM 2708 lub 2716, rozszerzając asortyment znaków do pełnego kodu ASCII, obejmującego także małe litery.

```
;WYGASZANIE EKRANU

WYGAS: LXI   H,POCZEK   ;ADRES POCZ. PAMIECI EKRANU
       LXI   B,800H     ;2048 POZYCJI
WYG1:  MVI   M,20H      ;WPIS SPACJI
       DCX   B          ;DEKREMENTACJA LICZNIKA
       MOV   A, B       ;CZY LICZNIK ZERO?
       ORA   C
       JNZ   WYG1       ;NIE ZERO — PETLA


       ;WPIS ZNAKÓW ASCII

PISZ:  LXI   H,POCZEK   ;ADRES POCZ. PAMIECI EKRANU
       MVI   B,40H      ;64 ZNAKI
       MVI   C,AOH      ;SPACJA + KURSOR
       MOV   M,C        ;WPIS KURSORA
       MVI   A,20H      ;KOD PIERWSZEGO ZNAKU
PISZ1: MOV   M,A        ;WPIS ZNAKU
       INR   A          ;NASTEPNY KOD
       INX   H          ;NASTEPNA POZYCJA WPISU
       MOV   M,C        ;WPIS KURSORA
       DCR   B          ;CZY WSZYSTKIE ZNAKI?
       JNZ   PISZ1      ;NIE WSZYSTKIE PETLA
       END
```

Na wydruku przedstawiono prosty program obsługi opisywanego sterownika, realizujący wstępne wygaszenie całego ekranu, a następnie wpisanie 64 znaków ASCII z jednoczesnym przesuwaniem kursora na pozycję następnego znaku.

GRZEGORZ STĘPIEŃ
MIROSŁAW DOLEŻYCH

IBSPiE, Politechnika Warszawska

● Klub mikrokomputerowy ABAKUS zmienił siedzibę. Z dalekiej Sadyby przeniósł się do śródmiejskiego Centrum Gier i Rozrywek. W nowym lokalu na razie nie jest zbyt „rozrywkowo", bo przecieka dach i kapie woda. Jednak nie zważając na trudności wznowiono działalność i komputery są już dostępne dla klubowiczów. Niezrzeszeni będą mogli „pomacać" komputer raz w tygodniu w tzw. dzień otwarty. Jako nowość przewidziano spotkania dla dzieci w wieku od 2 do 4 lat. Nie jest to zapowiadany w mass-mediach żłobek mikrokomputerowy, bowiem rodzice muszą sami zmieniać pieluchy.

Obecny adres klubu: Al. Jerozolimskie 2, Warszawa (naprzeciwko Muzeum Narodowego) tel. 27-87-73 w. 3.

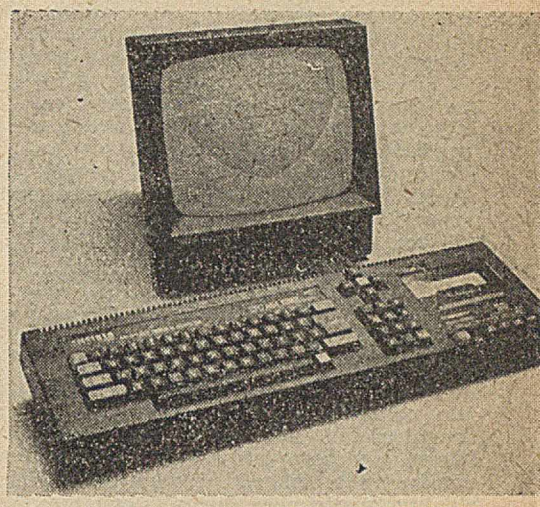
● „ABAKUS" ogłosił kolejny konkurs na grę edukacyjną. Poszukiwane są programy łączące elementy zabawy z przekazywaniem wiedzy lub wyrabianiem pewnych umiejętności (ale nie mięśni lub refleksu). Gra może być adresowana do osób w dowolnym wieku. Nie przewidziano również żadnych ograniczeń w zakresie przekazywanej wiedzy. Wysoko oceniane będą programy uniwersalne — czyli takie, które można wykorzystać dla szerokiej klasy zagadnień, przy niewielkich modyfikacjach.

Program powinien być opracowany dla jednego z dostępnych w klubie mikrokomputerów (COMMODORE C64, ZX SPECTRUM, ZX 81, ORIC 1, lub TI 99/4). Nie wyklucza się możliwości zgłaszania programów przeznaczonych dla innych mikrokomputerów, jednak w tym przypadku autor zobowiązany jest do zaprezentowania programu razem z komputerem (w uzgodnionym z jury terminie).

Programy (na taśmie lub dyskietce) powinny wpłynąć do klubu przed 30 listopada 1985 roku. Po zakończeniu konkursu nie przewiduje się odsyłania nośników (staną się one własnością klubu). Klub zastrzega sobie również prawo do wykorzystywania nadesłanych programów w działalności klubowej (oczywiście bez prawa rozpowszechniania).

Wśród nagród przewidziano m.in. COMMODORE C64, TI 99/4 oraz 50 gier komputerowych.

Opisujemy komputer, który wprowadzono na rynek bez fanfar i bicia dzwonów, jak to miało miejsce przy pojawieniu się SINCLAIRA QL czy japońskich MSX-ów. Początkowo niedostrzeżony, znika obecnie ze sklepów jak przysłowiowa kamfora. Można przewidywać, że niebawem zrobi nie mniejszą karierę niż ZX SPECTRUM; wielkim jego atutem jest niska cena (po odliczeniu ceny monitora TV — zaledwie o 10 funtów większa od ceny ZX SPECTRUM). Na razie polecamy tę konstrukcję opierając się jedynie na literaturze. Niebawem spróbujemy przedstawić porównanie podanych rewelacji z praktycznymi doświadczeniami.



AMSTRAD CPC 464

WOJCIECH JURZYK

Jednym z ciekawszych komputerów domowych, jakie pojawiły się pod koniec ubiegłego roku, jest CPC 464, angielskiej firmy AMSTRAD. Jego konstrukcja nie jest rewelacyjna, uniknięto natomiast niezbyt szczęśliwych rozwiązań stosowanych w innych mikrokomputerach, dając w zamian istotnie lepsze.

Ewenementem jest fakt, że komputer ten jest sprzedawany wraz z monitorem. Użytkownik ma dwie możliwości do wyboru: monitor 14" kolorowy lub monochromatyczny z zielonym ekranem. Większego wyboru nie ma, jako że komputer zasilany jest właśnie z monitora. Dla tych, którzy nie chcą monitora, proponuje się osobny zasilacz wraz z modulatorem do stadardowego odbiornika TV (w systemie PAL), gwarantując jednocześnie dobrą jakość obrazu.

Zaskakujący jest fakt zintegrowania magnetofonu z klawiaturą i procesorem w jednej obudowie. Magnetofon jest zaopatrzony w licznik obrotów i częściowo sterowany z programu. Programista ma możliwość wyboru szybkości zapisu: o podwyższonej niezawodności 1000 lub 2000 bodów (jak wynika z materiałów producenta, nawet ta szybsza zapewnia zadowalającą jakość). Szybkość odczytu wybierana jest automatycznie.

Najistotniejszą niemal częścią komputera jest dla użytkownika klawiatura, z nią bowiem ma on najwięcej do czynienia. I tu AMSTRAD CPC 464 oferuje rozszerzone — w stosunku do komputerów domowych — możliwości; oprócz standardowej co do wielkości i rozkładu klawiszy (QWERTY) „prawdziwej" klawiatury jest 17 dodatkowych klawiszy — 5 przeznaczonych na sterowanie kursorem oraz 12 tworzących klawiaturę numeryczną. Zastrzeżenia budzi natomiast rozkład dodatkowych klawiatur — przy posługiwaniu się kursorem łatwo może dojść do przypadkowego naciśnięcia klawiszy numerycznych.

Użytkownik może również zaopatrzyć się w napęd dysków elastycznych. Jeżeli do tego doda się fakt, że można wtedy pracować pod kontrolą systemu operacyjnego CP/M, zaskoczenie rośnie. Na dyskietkach można mieszać zarówno pliki utworzone pod kontrolą CP/M, jak i AMSDOS (wewnętrzy system operacyjny). Możliwe są transmisje w obu kierunkach tzn. z magnetofonu na dysk i odwrotnie.

Wewnętrzną budowę CPC 464 cechuje brak ekstrawagancji — zastosowano procesor Z80A (istotne dla polskich użytkowników szkolonych na INTELU 8080) z częstotliwością zegara 4 MHz. Pamięć RAM 64 KB i do tego 32 KB pamięci ROM stronicowej po 16 KB i umieszczonej pod tymi samymi adresami co pamięć obrazowa (też 16 KB).

Użytkownik może wybrać jeden z trzech typów: 20, 40 lub 80 znaków w każdym z 25 wierszy i odpowiednio 2, 4 i 16 kolorów (z palety 27) dla k a ż d e g o punktu na ekranie. Z materiałów informacyjnych wynika, iż tryb o najwyższej rozdzielczości daje dobrą jakość tekstów jedynie na monitorze monochromatycznym (tekst w kolorze jest nadal czytelny, ale już nie tak wyrazisty).

Kilka z cech sprzętowej konstrukcji wymaga jednak szerszego opisu.

Po pierwsze — stronicowanie pamięci. Przewidziane jest dołączenie do 240 stron, z których odczyt jest prosty, natomiast ewentualny zapis nieco bardziej skomplikowany, choć także możliwy. Jest to więc prosty sposób na poszerzenie możliwości komputera (dodatkowe pamięci ROM) bez konieczności zmniejszania dostępnej pamięci RAM.

Po drugie — sterownik CRT i procesor są tak zsynchronizowane, że ich dostęp do pamięci nie następuje jednocześnie. Oznacza to, że procesor nie może wykonać więcej niż trzech cykli maszynowych bez stanu WAIT. Tu jednak autorzy oprogramowania (firma LOCOMOTIVE SOFTWARE) popisali się sprawnością — tak je konstruując, aby jak najmniej stracić z tego powodu na szybkości. AMSTRAD CPC 464 jest tylko nieco wolniejszy od BBC/Acorn.

Użytkownik ma do dyspozycji standardowe łącze Centronics dla dołączenia drukarki oraz osobno wyprowadzenie pełnej szyny procesora Z80 z kilkoma dodatkowymi sygnałami, np. do dołączenia pióra świetlnego — w sumie 50 sygnałów, wszystkie zbuforowane. Ponadto istnieje możliwość dołączenia drążka sterowniczego (a następnie do niego drugiego).

Dziwić może nieobecność niezależnego przycisku RESET — zerowanie realizowane jest na zasadzie trzech klawiszy: SHIFT, CONTROL, ESCAPE.

Do pełnego opisu „kontaktu ze światem" dodać trzeba jeszcze informacje o możliwościach dźwiękowych: trzy kanały siedmiooktawowe z szerokim zakresem sterowania wyprowadzane na wzmacniacz stereo lub bezpośrednio przez wewnętrzny głośnik komputera o regulowanym natężeniu dźwięku (słyszalnym jako mono).

Nie mniej godne uwagi jest zaimplementowane w AMSTRADZIE CPC 464 oprogramowanie — zarówno w części dotyczącej interpretera języka BASIC, jak i systemowej. BASIC oparty jest na standardzie przemysłowym opracowanym przez znaną firmę MICROSOFT, a dodatkowo zawiera w sobie dużo nowych funkcji do sterowania dźwiękiem, grafiką, wejściem—wyjściem, redagowaniem programów oraz (uwaga!) przerwaniami czasu rzeczywistego.

Z nowym rozwiązaniem spotykamy się już w edytorze. Oprócz standardowego edytora wierszowego, mamy do czynienia z kursorem kopiującym, dzięki któremu tekst może być kopiowany z jednego miejsca ekranu na inne. Istotnymi pomocami przy uruchamianiu programu w języku BASIC są: tryb śledzenia oraz instrukcja ON ERROR. Program można zatrzymać naciskając klawisz ESCAPE, zaś przerwać kolejnym naciśnięciem tegoż klawisza. Podobnie, tryb automatycznego przenumerowania linii programu jest dla użytkownika istotną pomocą.

Podstawowy zestaw instrukcji języka BASIC został poszerzony ponadto o takie instrukcje, jak WHILE...WEND oraz IF...THEN...ELSE, zaś instrukcja PRINT USING znacznie ułatwia pracę przy formatowaniu informacji wyjściowej.

Zdecydowanie większe niż w innych mikrokomputerach są możliwości pro-

gramowania grafiki. Standardowy zbiór instrukcji DRAW, PLOT, MOVE (przesunięcie kursora bez śladu) i TEST (podaje kolor punktu na ekranie) występuje również w postaci relatywnej, zgodnej z nowym układem współrzędnych definiowanych w instrukcji ORIGIN. Można też wyprowadzać teksty na obrazy graficzne. Pracując w języku BASIC, użytkownik ma do dyspozycji 42 KB RAM.

Kolejne dwie cechy oprogramowania: to ostatni krzyk mody w tej branży: okna i strumienie (ang. windows & streams). Można tu zdefiniować osiem okien, z czego jedno graficzne, i mogą one się nawzajem nakładać. Okna powiązane są ze strumieniami jako ich obiekty wyjściowe, podobnie jak drukarka i magnetofon lub napęd dyskowy. Jako obiekty wejściowe służą: klawiatura, magnetofon i dysk. Obsługa strumieni jest wyjątkowo prosta, ale poważną wadą jest zerowanie wszystkich okien przy zmianie trybu pracy (np. z 40 na 80 kolumn).

Cechą niezwykle rzadko spotykaną w tego typu konstrukcjach są przerwania. CPC 464 ma cztery wewnętrzne zegary dostępne dla programisty, które mogą być użyte do generowania przerwań. Dotychczas przerwaniami zajmowali się konstruktorzy i programiści rozwiązujący zagadnienia typu systemowego, zaś za pomocą AMSTRADA CPC 464 każdy może odbyć „kurs programowania" w czasie rzeczywistym. Instrukcja AFTER generuje pojedyncze uprzednio zdefiniowane przerwanie, zaś EVERY czyni to w regularnych odstępach czasu.

Autorzy oprogramowania dołożyli dużo starań, aby ułatwić użytkownikowi dostęp do oprogramowania firmowego. Jego pełny opis znajduje się w odpowiednim podręczniku. Producent z zasady stara się udostępnić użytkownikom wszelkie informacje mogące służyć lepszemu wykorzystaniu lub poszerzeniu możliwości mikrokomputera.

Reasumując — AMSTRAD CPC 464 jest przykładem dobrego rozwiązania pod względem technicznym — zarówno od strony sprzętowej, jak i programowej. Niemal grzechem byłoby używać tego komputera do tak prymitywnych zastosowań jak gry. Przecież to prawie profesjonalny sprzęt: wygodna klawiatura, 80-kolumnowy ekran, dysk elastyczny pracujący pod systemem CP/M (dostarczanym wraz z językiem LOGO — wersja opracowana przez firmę DIGITAL RESEARCH specjalnie dla AMSTRADA CPC 464) oraz bardzo tani „backup" — w polskich warunkach cechy o niebagatelnym znaczeniu. Można się w krótkim czasie spodziewać dużej ilości oprogramowania na kasetach i dyskach (do końca 1984 roku oferowano ponad 200 tytułów).

Wielu producentów oprogramowania dostosowuje swoje programy (napisane dla innych komputerów) dla CPC 464, nie wspominając już o niezliczonych zasobach programów pracujących pod nadzorem CP/M.

MAREK GÓRECKI
Warszawa

Akademia mikroKLANU stara się uczyć elegancji w pisaniu programów. Jej lekcje przeznaczone są dla osób mających podstawowe doświadczenie w programowaniu. Liczymy na propozycje tematyczne od „uczniów".

Akademia mikroKLANU (5)

Wyrażenie

Kolejną część naszego cyklu poświęcimy pewnemu drobiazgowi... Spróbujmy wykonać program z wydruku 1. Jest to procedura mogąca być częścią dużego programu numerycznego.

```
10 A=0
20 B=1
30 C=2
40 A=A+B
50 B=SQR(1+A*A)
60 C=C+C
70 X=3*C/(A+B+A)
80 PRINT X
90 GOTO 40
```

Wartość X, powinniśmy skończyć obliczać gdy osiągniemy już pełną dokładność numeryczną komputera (patrz odcinek 1). Najbardziej prymitywną metodą jest utworzenie pętli explicite[1]) w tym celu dodajemy instrukcje:

```
35 FOR D=1 TO 8
90 NEXT D
```

Po poprawkach procedura wydrukuje osiem wartości (zbieżnych do ..., trzeba się samemu przekonać do czego!) i zakończy się.

```
34 X=0
35 EPS=1/1000000
36 Y=X
80 IF ABS(X-Y)>EPS THEN GOTO 36
90 PRINT X
```

Gdy dysponujemy komputerem o dużej dokładności obliczeń wolelibyśmy jednak wyliczać wartość X — aż do otrzymania pożądanej dokładności. Zatem niech EPS oznacza żądaną dokładność bezwzględną. Wtedy, wracając do pierwotnej wersji procedury i dokonując poprawek jak na wydruku 2, możemy spokojnie (podstawiając różne wartości EPS) obliczać wartość X. Jeśli algorytm obliczania naszego wyrażenia jest skończony (już raz mieliśmy do czynienia z tym problemem przy Zasadzie Ulama — odcinek 3), to zawsze otrzymamy odpowiednią liczbę z żądaną dokładnością. Może to jednak trwać zbyt długo, jeżeli X jest duże, a zbieżność procedury powolna (EPS wybraliśmy ambitnie małe!). Dlatego zawsze powinno się określić liczbę iteracji, po wykonaniu których — niezależnie od osiągniętej dokładności — program zakończy obliczenia. Zrobimy to wprowadzając modyfikacje z wydruku 3.

```
32 I=0
33 M=50
71 I=I+1
80 IF ABS(X-Y)>EPS AND I<M THEN GOTO 36
```

Jeszcze powinniśmy poprawić wyjście: nie zawsze potrzeba drukować wszystkie cyfry. Poza tym nie mamy komunikatu, że program skończył obliczenia z powodu wykonania 50 iteracji.

```
LIST

10 A=0
20 B=1
30 C=2
32 I=0
33 M=50
34 X=0
35 EPS=1/1000000
36 Y=X
40 A=A+B
50 B=SQR(1+A*A)
60 C=C+C
70 X=3*C/(A+B+A)
71 I=I+1
80 IF ABS(X-Y)>EPS AND I<M THEN GOTO 36
90 IF I=M THEN PRINT "WYKONANO 50 ITERACJI"
100 X0=INT(X/EPS+0.5)*EPS
110 PRINT "X=";X0
READY.
```

Tak więc prosta procedura nieco się rozrosła (wydruk 4) — za to spokojnie możemy umieścić ją jako przetestowany blok naszego superprogramu!

Pozostaje pytanie: dlaczego ten program daje właśnie taki rezultat. Nie wiemy tego — może Czytelnicy potrafią zrekonstruować algorytm?

JAKUB TATARKIEWICZ

[1]) Przykład zaczerpnięty z POPULAR COMPUTING, maj 1984

prowadzi:
Andrzej J. Piotrowski
tel. dom.: 48-22-85